# Computational aspects of simplex and MBU-simplex algorithms using different anti-cycling pivot rules

Tibor Illés[†,*], Adrienn Nagy[‡,†]

October 8, 2012

**Abstract**

Several variations of index selection rules for simplex type algorithms for linear programming, like the Last-In-First-Out or the Most-Often-Selected-Variable are rules not only theoretically finite, but also provide significant flexibility in choosing a pivot element. Based on an implementation of the primal simplex and the monotonic build-up (MBU) simplex method, the practical benefit of the flexibility of these anti-cycling pivot rules are evaluated using public benchmark LP test sets. Our results also provide numerical evidence that the MBU-simplex algorithm is a viable alternative to the traditional simplex algorithm.

*Keywords:* monotonic build-up simplex algorithm, primal simplex method, index selection rules, linear programming.

*AMS Mathematics Subject Classification*: 90C05

## 1 Introduction

Different practical implementations of the revised primal and dual simplex methods still form one of the basis of most mathematical programming applications. Several different simplex algorithm variants and different pivot selection rules have been invented [22], but with the exception of problem specific alternatives like the network simplex method, the alternative methods have not gained similar importance, and are generally not widely available.

To ensure finiteness for degenerate problems, a large number of well know index selection rules are available. While many of these rules are mainly of theoretical interest, there is a significant number [7] of flexible index selection rules [1], claiming practical applicability by offering various degrees of flexibility while still providing theoretical finiteness [6].

Traditional index selection rules are typically not applied directly in implementations of the simplex methods, as in case of stalling progress is most often achieved by some means of perturbation [14]. Also, it is difficult to provide solid numerical evidence to the practical benefits of index selection rules. The reasons are many fold:

[*†]Email: illes@math.bme.hu, Department of Differential Equations, Technical University Budapest

[†‡]Corresponding author. Email: adriennnagy@fico.com, Research carried out at Department of Operations Research, Eötvös Loránd University, Budapest

- implementations have to either compete against well established commercial and academic solvers with several years of work hours invested in them,
- or alternatively are based on own implementations where the implementation efforts are often dominated by numerical challenges,
- efficiency of the implementations tend to be a function of method specific algorithmic features (like shifting for the primal method, or bound flipping for the dual method, quality of the basis factorization and updates) making direct comparisons of the algorithms measure the depth of implementations instead of the algorithms themselves.
- cycling is a relatively rare phenomenon as often claimed, see [5](chapter 3, Pitfalls and How to Avoid Them) and [16](chapter 9.4, Do computer codes use the lexico minimum ratio rule?).

In consequence, numerical results using different implementations are hard to judge and to be used as a reference (for a detailed description on implementing the simplex methods see Maros's book [14]).

The purpose of this paper is to provide numerical evidence that the monotonic build-up (MBU) simplex method [2] is a viable alternative to the tradition primal simplex method, and to provide a measure of the advantages of the use of flexible index selection rules.

Section 2 of the paper states the form of the linear programming problem used and summarizes the algorithmic results the paper is built on: the traditional revised simplex method and the MBU-simplex method, the s-monotone index selection rules including the Last-In-First-Out (LIFO) and the Most-Often-Selected-Variable (MOSV) rules and some of their generalizations that make use of the flexibility provided.

Section 3 presents how we propose to address the difficulties in comparing different pivot methods and index selection rules. The main highlights are

- the standard form of the linear problems is used,
- the tests are carried out on public benchmark sets which are transformed to the standard form,
- only such algorithmic features are used that are generally applicable to all of the investigated algorithms,
- the numerical linear algebra used is based on the API of a well established linear programming solver,
- the implementations are made available for reference and reproducibility.

Section 4 provides details of the implementations and the different specifications on how the advantage of flexibility in the index selection rules is measured.

In section 5 our numerical results and the derived conclusions are presented.

The paper is closed by a summary.

### 1.1 Notations

Throughout this paper, matrices are denoted by italic capital letters, vectors by bold, scalars by normal letters and index sets by capital calligraphic letters. Columns of a matrix are indexed as a subscript while rows are indexed by superscripts. $A$ denotes the original problem matrix, $\mathbf{b}$ the right hand side and $\mathbf{c}$ the objective vector. For a given basis $B$, we denote the nonbasic part of $A$ by $N$; the corresponding set of indices for the basis and nonbasic part are denoted by $\mathcal{I}_B$ and $\mathcal{I}_N$ respectively. The corresponding short pivot tableau for $B$ is denoted by $T := B^{-1}N$, while the transformed right hand side and objective is denoted by $\overline{\mathbf{b}} := B^{-1}\mathbf{b}$ and $\overline{\mathbf{c}} := \mathbf{c}^T B^{-1}$. The algorithms in this paper will refer to the rows and columns of the short pivot tableau corresponding to a given basis $B$ as $\mathbf{t}_j := B^{-1}\mathbf{a}_j$ and $\mathbf{t}^{(i)} := (B^{-1}N)^{(i)}$ [12].

## 2 Algorithms and pivot selection rules

The linear programming (LP) problem considers an optimization problem with a linear objective over linear equalities and inequalities. Most theoretical forms of the pivot algorithms work on the standard form of the problem - all constraints are equations and all variables have a uniform lower bound of zero and no upper bounds. Any LP problem can easily be converted to this form, although the practical effect of such a conversion can be significant. This standard form and its dual is

$$\begin{aligned} \min \mathbf{c}^T\mathbf{x} \\ A\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq 0 \end{aligned} \qquad\qquad \begin{aligned} \max \mathbf{y}^T\mathbf{b} \\ \mathbf{y}^T A \leq \mathbf{c} \end{aligned}$$

For the purpose of this study, we use the primal version of the simplex algorithm, as it is more comparable to the MBU method than the dual. The MBU-simplex method is not a dual method in the sense that it does not maintain dual feasibility, but instead it is complete when dual feasibility is achieved, just like in the case of the primal simplex. However, it is not strictly a primal method either, as primal feasibility is not maintain in every iteration, but rather it is restored every time the feasibility of a new dual variable is achieved. Still, these properties make the MBU arguably a primal like method, supporting the choice of selecting the primal simplex method for comparison.

Figure 1 and Figure 2 summarizes the simplex and the monotonic build-up simplex methods respectively. For both methods, a standard first phase using artificial slacks is used. The key steps are numbered, and are marked to match in the pseudo code as well.

*The primal simplex method*

**input data:**
$A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n$, $\mathcal{I} = \{1, \ldots, n\}$;
The problem is given in the canonical form: $\min \mathbf{c}^T\mathbf{x}, A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0$;
$B$ feasible basis, $\mathcal{I}_B$ its index set, $\overline{\mathbf{b}} := B^{-1}\mathbf{b}$, the current basic solution;
**begin**
calculate $\overline{\mathbf{c}} := \mathbf{c} - (\mathbf{c}_B^T B^{-1})A$, the reduced costs;
$\mathcal{I}_- := \{i \in \mathcal{I}_N \,|\, \overline{c}_i < 0\}$, indices of the non optimal columns; (1)
**while** $(\mathcal{I}_- \neq \emptyset)$ **do**
choose $q \in \mathcal{I}_-$ according to the used index selection rule; (3)
calculate $\mathbf{t}_q := B^{-1}\mathbf{a}_q$;
**if** $(\mathbf{t}_q \leq 0)$
**then** STOP: problem dual infeasible; (5)
**else**
let $\vartheta := min\left\{\frac{\overline{b_i}}{t_{iq}} | i \in \mathcal{I}_B,\ t_{iq} > 0\right\}$; (primal ratio test) (4)
choose $p \in \mathcal{I}_B$ arbitrary,
where $\frac{\overline{b_p}}{t_{pq}} = \vartheta$ according to the used index selection rule;
**endif**
pivoting; update $B^{-1}$ and $\overline{\mathbf{b}}$; (6)
check need for re-inversion; (7)
update $\overline{\mathbf{c}}$ and $\mathcal{I}_-$;
**endwhile**
STOP: An optimal solution is found; (2)
**end**

Figure 1: The primal simplex method.

The key steps of the algorithm are

1. Determine the possible incoming column index set $I_-$ which contains those variables for which the reduced costs are negative.
2. If the $I_-$ set is empty, there are no more improving columns and the algorithm terminates.
3. Otherwise use the *index selection rule* to select an improving column.
4. Perform a primal ratio test to determine the step size, and select the outgoing column using the *index selection rule*.
5. If there are no suitable pivot elements then the problem is dual infeasible.
6. Carry out the pivot, update the basis and the factorization.
7. Check if re-inversion is necessary, and start over again.

A significant advantage of the primal simplex method is its simplicity.

In comparison, the monotone build-up simplex method is relatively complex, as the design to maintain monotonicity in the feasibility of the dual variables (reduced cost) requires an extra dual ratio test in each iteration. The monotonic build-up simplex algorithm selects an infeasible dual variable (called the driving column), and works to achieve its feasibility while keeping all the already dual feasible variables dual feasible.

*The monotonic build-up (MBU) simplex algorithm*

**input data:**
$A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n, \mathcal{I} = \{1, \ldots, n\}$;
The problem is given in the canonical form: $\min \mathbf{c}^T\mathbf{x}, A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0$;
$B$ feasible basis, $\mathcal{I}_B$ its index set, $\overline{\mathbf{b}} := B^{-1}\mathbf{b}$, the current basic solution;
**begin**
calculate $\overline{\mathbf{c}} := \mathbf{c} - (\mathbf{c}_B^T B^{-1})A$, the reduced costs;
$\mathcal{I}_- := \{i \in \mathcal{I}_N \mid \overline{c}_i < 0\}$, indices of non optimal columns; (1)
**while** $(\mathcal{I}_- \neq \emptyset)$ **do**
choose $s \in \mathcal{I}_-$, the driving variable according to the used index selection rule; (3)
**while** $(s \in \mathcal{I}_-)$ **do**
calculate $t_s := B^{-1}\mathbf{a}_s$;
let $\mathcal{K}_s = \{i \in \mathcal{I}_B \mid t_{is} > 0\}$;
**if** $(\mathcal{K}_s = \emptyset)$
**then** STOP: problem dual infeasible; (5)
**else**
let $\vartheta := min\left\{\frac{\overline{\mathbf{b}_i}}{t_{is}} \mid i \in \mathcal{K}_s\right\}$; (the primal ratio test) (4)
choose $r \in \mathcal{K}_s$ according to the used index selection rule, where $\frac{\overline{\mathbf{b}_r}}{t_{rs}} = \vartheta$
and let $\theta_1 := \frac{|\overline{\mathbf{c}_s}|}{t_{rs}}$;
Let $\mathcal{J} = \{i \in \mathcal{I} \mid \overline{\mathbf{c}_i} \geq 0 \text{ and } t_{ri} < 0\}$;
**if** $(\mathcal{J} = \emptyset)$ **then** $\theta_2 := \infty$;
**else**
$\theta_2 := min\left\{\frac{\overline{\mathbf{c}_i}}{|t_{ri}|} | i \in \mathcal{J}\right\}$, (the dual ratio test) (6)
and choose $q \in \mathcal{J} : \theta_2 = \frac{\overline{\mathbf{c}_q}}{|t_{rq}|}$ according to the used index selection rule;
**endif**
**if**$(\theta_1 \leq \theta_2)$ (7)
**then** pivoting on the $t_{rs}$ element;
**else** pivoting on the $t_{rq}$ element;
**endif**
update $B^{-1}$ and $\overline{\mathbf{b}}$; (8)
check need for re-inversion; (9)
update $\mathbf{c}$ and $\mathcal{I}_-$ index set;
**endif**
**endwhile**
**endwhile**
STOP: An optimal solution is found; (2)
**end**

Figure 2: The monotonic build-up (MBU) simplex algorithm.

The key steps of the monotonic build-up simplex method are

1. If there is no active driving column, then determine the possible incoming column index set $I_-$ which contains the variables with negative reduced cost.
2. If the $I_-$ set is empty, there are no more improving columns and the algorithm is complete.
3. Otherwise use the *index selection rule* to select $s$ as the index of an improving column. This column will serve as the driving column, and it's value is monotonically increased until it becomes dual feasible.
4. Perform a primal ratio test on the positive elements of the transformed pivot column $\mathbf{t}_s$ of the driving variable and select the outgoing variable $r$ using the *index selection rule.*
5. If there are no suitable pivot elements then the problem is dual infeasible.
6. Using the row $\mathbf{t}^r$ selected by the primal ratio test, carry out a dual ratio test over $\mathcal{J}$ the dual feasible columns with negative pivot elements in row $r$, breaking ties using the *index selection rule.*
7. Compare the values of the two ratio test $\theta_1$ and $\theta_2$. If it's ratio is not larger, than choose the pivot $t_{rs}$ selected by the primal ratio test, otherwise choose the pivot $t_{rq}$ selected by the dual one.
8. Carry out the pivot, update the basis and the factorization.
9. Check if re-inversion is necessary, and iterate.

It is not straightforward why the MBU algorithm is correct as it is presented, so it is important to restate the following result:

**Theorem 1** *[2] Consider any pivot sequence of the MBU algorithm. Following a pivot selected by the dual side ratio test, the next basis produced by the algorithm has the following properties:*

1. $\bar{c}_s < 0$,
2. *if* $\bar{b}_i < 0$, *then* $t_{is} < 0$,
3. $\max\left\{\frac{\bar{b}_i}{t_{is}} \mid \bar{b}_i < 0\right\} \leq \min\left\{\frac{\bar{b}_i}{t_{is}} \mid t_{is} > 0\right\}$.

It is important to note that the proof of this results also shows that property *(3)* of the theorem holds for every basis produced by a pivot of the algorithm where the dual ratio test was smaller. Property *(2)* shows that whenever the primal ratio test is ill defined, the problem is indeed unbounded, even if the current basis is primal infeasible.

**Theorem 2** *[2] When the MBU algorithm performs a pivot selected by the primal side ratio test on $t_{rs}$, then the next basis is primal feasible.*

Note that if the selected pivot is primal non-degenerate then the objective function strictly increases, providing progress and finiteness.

Most commercial solvers employ some kind of a greedy approach in selecting an incoming variable instead of an index selection rule; like steepest edge. Also, as long as the objective improves, no index selection is necessary. In case of degeneracy, they typically perturb the problem. One of the sources of cycling could be coming from removing this perturbation, which cause the right hand side to change, needing some clean up iterations. This is a situation when cycling might occur, and when index selection rules might have a very practical role.

### 2.1 Flexible index selection rules

To ensure finiteness of the algorithm in the presence of degenerate pivots, we use flexible index selection rules. As a common framework of these rules, we use the concept of s-monotone index selection rules introduced in [7]. This framework defines a preference vector $\mathbf{s}$ for each index selection rule, called the primary rule. In a tie situation when the algorithm has to select an index among a set of candidates, it picks the one with the maximum value in $\mathbf{s}$. If this value is not uniquely defined, the algorithm can select any index with maximal $\mathbf{s}$ value (this freedom is the flexibility of the rule), while still preserving finiteness. The choice of index in these situations will be based on a secondary rule, aiming the benefit from the provided flexibility.

The following finite [7] s-monotone primary rules are used in our paper, which we present with the associated update of the $\mathbf{s}$ vector.

- *Minimal index/Bland:* The variables with the smallest index is selected.
  Initialize $\mathbf{s}$ consisting of constant entries

  $$s_i = n - i, i \in \mathcal{I}$$

  where n is the number of the columns in the problem.

- *Last-In-First-Out (LIFO):* The most recently moved variable is selected.
  Initialize $\mathbf{s}$ equal to $\mathbf{0}$. For a pivot at $(k, l)$ in the $r^{th}$ iteration update $\mathbf{s}$ as

  $$s_i^{'} = \begin{cases} r & \text{if } i \in \{k, l\} \\ s_i & \text{otherwise} \end{cases}$$

- *Most-Often-Selected-Variable (MOSV):* Select the variable that has been selected the largest amount of times before.
  Initialize $\mathbf{s}$ equal to $\mathbf{0}$. For a pivot at $(k, l)$ in the $r^{th}$ iteration update $\mathbf{s}$ as

  $$s_i^{'} = \begin{cases} s_i + 1 & \text{if } i \in \{k, l\} \\ s_i & \text{otherwise} \end{cases}$$

These primary pivot rules are used in combination with the following two secondary rules:

- *Bland's or the minimal index rule:* select the variable with the smallest index.
- *Dantzig's or the most negative rule:* select the variable with the smallest reduced cost.

The flexibility of the LIFO and MOSV rules will be used by the secondary rule. Naturally, benefits can be expected from the use of Dantzig's rule, and we will refer to these as

- *Hybrid-LIFO:* when the LIFO is not unique, select the one according to the most negative rule.
- *Hybrid-MOSV:* when the MOSV is not unique, select the one according to the most negative rule.

In this paper, we have considered all combinations of the above summarized index selection rules and the primal simplex and the MBU-simplex methods.

# 3 Implementation details and problem sets

Our goal is to compare the properties and efficiency of the selected pivot methods with the flexible index selection rules in a suitable numerical environment. For this purpose, we work on the standard form of the problems and without applying a presolver - which would have undone the conversion to the standard form anyway.

We aim to provide a framework for the comparisons, where the comparison is based on the main algorithmic features and the index selection rules, not between the depth of the implementation. In order to provide the uniform test environment for different algorithms and their variants depending on the anti-cycling pivot rules in our implementations, the basic version of the algorithms is implemented without further computational techniques that usually accelerate the computations, but often are specific to the algorithm in question, and would thus make a fair comparison difficult. To implement such improvements for both algorithms in an appropriate and equally efficient way is a nontrivial, challenging and interesting task.

The algorithms are initiated from a slack basis, and the traditional first phase method is used to produce an initial feasible bases for both algorithms.

One of the crucial numerical properties of most simplex methods is to maintain monotonicity in the selected (ether primal or dual) form of feasibility. In the presence of numerical error, one of the most widely applied methods to address infeasibility occurring due to numerical error is shifting. Shifting removes numerical infeasibility by rounding small negative numbers to zero in the transformed right hand side. In turn, to remove the side effects when re-inverting (re-introducing infeasibleness and change in the objective due to changes in the solution can result in breaking the monotonicity of the objective that in turn can cause cycling if not addressed in other way even in the presence of the index selection rules), the original right hand side is also shifted by adding the original column of the shifted variable to the non-transformed right hand side by the same amount. This perturbation has to be removed at the end, and a clean up phase applied. Our implementation does not apply this technique.

On the other hand, maintaining the concept of dual monotonicity in the monotonic build-up simplex method has proven to be crucial for its performance. However, the required monotonicity can be achieved by simply not allowing to chose a column twice as a driving variable, even if the round of errors make an feasible dual value infeasible again (relative to the selected optimality tolerance).

To achieve a fair comparison, problems where these rules were causing significant deviance from feasibility or optimality have been removed from the test sets.

To address scaling, most solvers automatically re-scale the problems. We have tested the algorithms on the transformed (to standard from) original and on a re-scaled version of the test sets. We used the exposed scaling functionality of the Xpress solver, and has observed an increase of 2% in success rate on the selected test set. As the improvement was not significant, we opted not to scale the problems.

To create a comprehensive, readily accessible and medium difficult set of test problems, we have used a selection of problems from the following 3 databases:

1. NETLIB [18]
2. Miplib 3 [15]
3. Miplib 2010 [15]

From among the Miplib collections, we only considered the linear part of the problem.

Benchmark sets are typically compiled from problems that are either computationally hard and/or otherwise complex and interesting; and while they provide suitable grounds for testing they set a very high requirement standard. Some of the NETLIB problems are known to be numerically challenging [19] - especially in the absence of presolve - and while the Miplib sets were created to be difficult or large integer problems, their relaxations are typically easier to solve. In section 4.3, numerical results are presented for a separate selection of industrial problems taken from the literature [13].

The selected problems were converted to the standard form. The average increase in the number of rows, columns and non-zeros is shown in Table 1:

| | |
|---|---|
| Number of extra rows | +2139% |
| Number of extra columns | +143% |
| Number of extra nonzeros | +27% |

Table 1: The average increase in problem size as a result of the conversion to standard form.

The very large increase in terms of row numbers is down to a relatively few problems that are flat (have a much larger number of columns than rows and the columns are both upper and lower bounded which is typical for several binary problems. On problems with both lower and upper bounds, the conversion to the standard from introduces $+n$ new rows, which can be a significant increase if $n \gg m$ where $n$ is the number of original columns and $m$ is the number of original rows).

As a note, Xpress itself takes 39% longer on average to solve the standard forms with presolve off (the choice of the primal algorithm is important in this respect, as converting the variables with both lower and upper bounds affects dual more through the missed opportunities of bound flips than primal, while the primal ratio test would need to consider both bounds anyway).

The criteria for a problem to be included was the following:

1. all algorithm and index selection rule combination solved the problem successfully,
2. the optimal objective matched the value reported by Xpress,
3. Xpress was able to solve the problem within 5 minutes,
4. a time limit of 1 hour was used for Netlib, and 5 minutes for the Miplib datasets.

Using this criteria, 108 problems were selected, with the average problem statistics presented in Table 2. The 11 assembly line balancing and workforce skill balancing problems (Balancing) from [4] and [13] were treated as a separate set.

| | Total size | Selected | average rows | average columns | average density |
|---|---|---|---|---|---|
| Netlib | 98 | 43 | 505 | 1082 | 2.07% |
| Miplib3 | 64 | 28 | 660 | 1153 | 1.20% |
| Miplib2010 | 253 | 37 | 1493 | 2500 | 0.63% |
| Balancing | 11 | 11 | 279 | 470 | 3.03% |

Table 2: The average size statistics of the selected test problems. The average values are calculated on the selected subset of problems, using the standard form.

In order to minimize the effect of the underlying linear algebra, we used the callable library of a commercial solver. We implemented our programs in the C programming language, using the XPRS function class of Xpress (Xpress is free for academic use). We used functions for the data retrieval, loading the basis, the back and the forward transformation, pivoting, and to get the pivot order.

The source code of our implementation is available online at [20].

# 4 Numerical results

There is a total of 108 problems in the selected test set. However, it is interesting to see how large the selected test set would be if only either the simplex or the MBU algorithm used in the selection process.

| | All versions solved | Number of extra wins |
|---|---|---|
| Simplex | 131 | 23 (+21%) |
| MBU | 118 | 10 (+9.2%) |
| Either MBU or Simplex | 141 | 33 (+30.5%) |

Table 3: Number of problems solved across all index selection rules using either the simplex or the MBU algorithm.

The fact that our simple implementation of the MBU managed to solve fewer problems is not surprising, as it is a more complex algorithm than the traditional primal simplex and as such with more opportunities for numerical issues. It is notable how large the number of problems discarded from the selection due to only one of the algorithms is, and clearly indicates the very different solution path the algorithms take. It is also notable that even though the traditional simplex method is more stable, it is not dominating the MBU variant.

The complete solution statistics is presented in Table 4:

| | **Most negative** | **Minimal** | **LIFO** | **MOSV** | **Hybrid LIFO** | **Hybrid MOSV** |
|---|---|---|---|---|---|---|
| **Simplex** | 209 | 150 | **168** | 150 | 161 | **175** |
| **MBU** | 150 | 145 | 149 | 149 | 146 | 146 |

Table 4: Total number of problems solved by algorithm and index selection rule combinations on all candidate problems.

The highest success rate is achieved by the simplex method with the Dantzig index selection rule. The result using this method is provided as a reference only, as it is not a theoretically finite method. As this test primarily measures numerical stability, the most robust method proved to be the relatively simpler one, doing the smaller number of iterations. Although this result is expected, its lead is larger than anticipated. It is also interesting that the Dantzig rule is less dominant in the case of the MBU; intuitively this is because the dual ratio test overwrites the original column selection. The question naturally arises: what selection rule (possibly greedy without the need of theoretical finiteness) would yield the best fit with the MBU method?

A selection of detailed results are presented in Table 5 for the NETLIB, and in Table 6 for the Miplib sets. In these tables, for each model and algorithm combinations, 3 numbers are presented: iteration, multiplicity and run time.

### 4.1 Iteration and time

In this section, all results refer to the selected 108 test problems.

Table 7 presents the fastest solution times among the simplex and the MBU-simplex algorithms, all timings rounded up to seconds (number of times a given algorithm was fastest with ties included in all).

Table 8 presents the total sums of iteration counts.

As expected, for the primal simplex, the most negative variable rule is the most efficient. From among the theoretically finite index selection rules, the Hybrid MOSV proved to be best. Although the MBU makes significantly less iterations, it does use more information in all iteration: the MBU spends more time per iteration, as it needs to calculate the transformed row

| | MBU Dantzig | MBU Minimal | MBU LIFO | MBU MOSV | MBU H-LIFO | MBU H-MOSV | Simplex Dantzig | Simplex Minimal | Simplex LIFO | Simplex MOSV | Simplex H-LIFO | Simplex H-MOSV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADLITTLE | 247 | 298 | 397 | 454 | 236 | 257 | 156 | 419 | 393 | 457 | 234 | 163 |
| | 1392 | 0 | 1830 | 1623 | 1002 | 1365 | 8734 | 0 | 6111 | 6308 | 5036 | 6300 |
| | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s |
| FIT1P | 3135 | 2824 | 2821 | 2821 | 2801 | 3626 | 3589 | 10407 | 7100 | 8022 | 9028 | 5507 |
| | 163797 | 0 | 885953 | 885953 | 171835 | 169080 | 2862086 | 0 | 1653008 | 1659394 | 1381221 | 1928092 |
| | 8s | 7s | 7s | 7s | 7s | 9s | 7s | 21s | 14s | 16s | 18s | 11s |
| MAROS-R7 | 17261 | 20857 | 12656 | 17183 | 21183 | 18031 | 4287 | 11434 | 14230 | 12320 | 22595 | 6300 |
| | 3355631 | 0 | 5697238 | 7431172 | 2560415 | 3514255 | 14601018 | 0 | 20624812 | 20893306 | 13668626 | 16756305 |
| | 251s | 312s | 175s | 238s | 308s | 257s | 43s | 111s | 137s | 120s | 246s | 65s |
| SCFXM1 | 1044 | 1261 | 985 | 981 | 1061 | 1030 | 630 | 5306 | 2760 | 4494 | 1986 | 1064 |
| | 35470 | 0 | 40217 | 42890 | 25459 | 29254 | 132223 | 0 | 141913 | 144835 | 80158 | 109563 |
| | 1s | 1s | 1s | 1s | 1s | 1s | 0s | 3s | 1s | 3s | 2s | 0s |
| SCORPION | 651 | 663 | 660 | 645 | 653 | 565 | 523 | 714 | 655 | 696 | 591 | 489 |
| | 31098 | 0 | 32285 | 31495 | 27225 | 29122 | 84824 | 0 | 60634 | 61300 | 63462 | 68268 |
| | 0s | 1s | 0s | 0s | 0s | 1s | 0s | 0s | 1s | 1s | 0s | 0s |
| SCSD1 | 767 | 1718 | 286 | 1953 | 1008 | 917 | 338 | 185742 | 4480 | 42395 | 671 | 460 |
| | 2057 | 0 | 43 | 1315 | 1031 | 2100 | 55034 | 0 | 79952 | 82081 | 44795 | 52397 |
| | 0s | 1s | 0s | 2s | 1s | 1s | 0s | 94s | 3s | 22s | 0s | 0s |
| SEBA | 2904 | 1776 | 1902 | 1852 | 3042 | 2808 | 2283 | 2239 | 2299 | 2333 | 2130 | 2462 |
| | 385775 | 0 | 429015 | 427110 | 271092 | 323841 | 1141102 | 0 | 751784 | 755741 | 639317 | 948853 |
| | 6s | 4s | 4s | 4s | 6s | 6s | 4s | 4s | 4s | 4s | 3s | 4s |
| SHELL | 2380 | 2300 | 2172 | 2423 | 2303 | 2160 | 1805 | 2414 | 2388 | 2257 | 2326 | 2025 |
| | 119134 | 0 | 138724 | 159986 | 57587 | 87425 | 991889 | 0 | 855983 | 883871 | 598344 | 813375 |
| | 6s | 6s | 5s | 7s | 6s | 6s | 4s | 4s | 4s | 4s | 4s | 4s |
| SIERRA | 6751 | 7478 | 6743 | 6476 | 6879 | 6613 | 5667 | 7216 | 7448 | 6887 | 6796 | 5626 |
| | 4417685 | 0 | 5226096 | 5234851 | 4350699 | 4482015 | 9503307 | 0 | 8289979 | 8326468 | 7989135 | 8218504 |
| | 43s | 48s | 43s | 42s | 44s | 42s | 27s | 34s | 35s | 33s | 32s | 26s |
| VTP-BASE | 573 | 486 | 499 | 518 | 589 | 561 | 494 | 489 | 570 | 526 | 616 | 552 |
| | 23435 | 0 | 33600 | 33747 | 20888 | 23157 | 89166 | 0 | 68299 | 69958 | 72766 | 73293 |
| | 0s | 0s | 0s | 0s | 1s | 0s | 0s | 1s | 0s | 1s | 1s | 0s |

Table 5: Numerical results on a set of NETLIB problems.

| | MBU Dantzig | MBU Minimal | MBU LIFO | MBU MOSV | MBU H-LIFO | MBU H-MOSV | Simplex Dantzig | Simplex Minimal | Simplex LIFO | Simplex MOSV | Simplex H-LIFO | Simplex H-MOSV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 416 | 505 | 538 | 502 | 474 | 442 | 266 | 1078 | 827 | 835 | 380 | 321 |
| bell3a | 10070 | 0 | 20636 | 20546 | 8940 | 10140 | 36278 | 0 | 47999 | 49317 | 34298 | 37170 |
| | 0s | 1s | 0s | 0s | 1s | 0s | 0s | 1s | 1s | 0s | 0s | 0s |
| | 11984 | 8515 | 8211 | 8680 | 15945 | 11915 | 11276 | 49770 | 38905 | 68400 | 66300 | 6374 |
| eilB101 | 4138209 | 0 | 4452538 | 4458614 | 4020427 | 4058276 | 22943568 | 0 | 4809496 | 4838383 | 11656561 | 12109399 |
| | 85s | 61s | 59s | 64s | 113s | 85s | 60s | 260s | 204s | 360s | 360s | 34s |
| | 2962 | 2771 | 2857 | 2916 | 3208 | 2916 | 4730 | 4155 | 4478 | 4507 | 5101 | 9989 |
| neos-1616732 | 86609 | 0 | 87594 | 87368 | 90364 | 87368 | 1695071 | 0 | 894758 | 894390 | 799322 | 1034908 |
| | 11s | 10s | 10s | 11s | 12s | 11s | 13s | 12s | 12s | 13s | 15s | 27s |
| | 17198 | 18872 | 28100 | 26097 | 16156 | 16733 | 16221 | 27614 | 28167 | 24722 | 17103 | 14467 |
| neos-555424 | 13083516 | 0 | 10522883 | 10444208 | 12902065 | 12917539 | 58089052 | 0 | 33859418 | 33747224 | 32763158 | 39078410 |
| | 221s | 245s | 360s | 339s | 215s | 217s | 161s | 271s | 279s | 246s | 172s | 144s |
| | 45 | 48 | 48 | 48 | 47 | 45 | 49 | 50 | 51 | 50 | 50 | 49 |
| flugpl | 418 | 0 | 441 | 441 | 376 | 382 | 861 | 0 | 769 | 770 | 632 | 730 |
| | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s | 0s |
| | 4977 | 8271 | 6845 | 7121 | 5361 | 5321 | 4531 | 8281 | 7488 | 8725 | 5610 | 4158 |
| gesa2 | 1598041 | 0 | 1795470 | 1944212 | 1409791 | 1483927 | 6582595 | 0 | 5705763 | 5651121 | 3339443 | 4002298 |
| | 21s | 37s | 29s | 31s | 23s | 24s | 15s | 26s | 24s | 28s | 19s | 13s |
| | 13083 | 11751 | 11884 | 11983 | 12125 | 13084 | 5675 | 44897 | 115709 | 208458 | 72643 | 29707 |
| go19 | 400003 | 0 | 381439 | 380045 | 407048 | 400386 | 3402593 | 0 | 973954 | 973515 | 714247 | 951911 |
| | 19s | 18s | 18s | 18s | 18s | 20s | 8s | 59s | 153s | 275s | 102s | 40s |
| | 3197 | 7370 | 5379 | 5256 | 2686 | 2448 | 3877 | 10241 | 7664 | 9452 | 4277 | 4385 |
| mik | 41538 | 0 | 63430 | 65280 | 29556 | 33851 | 1137428 | 0 | 193253 | 194068 | 183961 | 231635 |
| | 3s | 8s | 5s | 5s | 2s | 2s | 3s | 7s | 6s | 7s | 3s | 3s |
| | 998 | 864 | 1004 | 1113 | 902 | 762 | 866 | 2106 | 1516 | 1343 | 1270 | 1211 |
| p0201 | 41505 | 0 | 43162 | 45395 | 36531 | 38800 | 156780 | 0 | 104780 | 105236 | 102030 | 110295 |
| | 0s | 1s | 0s | 1s | 1s | 0s | 1s | 2s | 1s | 0s | 1s | 0s |
| | 583 | 505 | 514 | 520 | 549 | 569 | 690 | 635 | 553 | 572 | 772 | 644 |
| rgn | 16538 | 0 | 19613 | 19849 | 15601 | 16357 | 96388 | 0 | 54088 | 54428 | 51905 | 63232 |
| | 1s | 0s | 1s | 1s | 0s | 1s | 1s | 0s | 0s | 0s | 1s | 0s |

Table 6: Numerical results on a set of Miplib problems.

| | Most negative | Minimal | LIFO | MOSV | Hybrid LIFO | Hybrid MOSV |
|---|---|---|---|---|---|---|
| **Simplex** | 84 | 36 | 32 | 40 | 48 | **69** |
| **MBU** | 31 | 25 | 31 | 28 | 30 | 31 |

Table 7: Fastest solution time achieved in the selected test set.

| Iteration | Most negative | Minimal index | LIFO | MOSV | Hybrid LIFO | Hybrid MOSV |
|---|---|---|---|---|---|---|
| **Simplex** | 264 672 | 988 803 | 757 289 | 1 015 929 | 625 707 | **341 381** |
| **MBU** | 588 423 | 442 149 | 503 547 | 580 842 | 437 821 | **428 639** |

Table 8: The total sums of iteration counts.

as well (which is typically computationally significantly more expensive than the transformed column). However, even thought each iteration is more expensive, it seems to pay off in means of the average total time by doing fewer iterations. Table 9 presents the total solution times.

| Time | Most negative | Minimal index | LIFO | MOSV | Hybrid LIFO | Hybrid MOSV |
|---|---|---|---|---|---|---|
| **Simplex** | 1 017 | 2 626 | 2 286 | 2 744 | 2 275 | **1 100** |
| **MBU** | 2 269 | 2 086 | 2 312 | 2 498 | 2 083 | **2 036** |

Table 9: Total solution times.

The MBU seems to be the fastest in average time as well, so the extra investment per iterations pays off, although possibly not surprising, the faster strategy proved to be the most negative rule.

Our results indicate that:

- The most-negative rule with simplex is the fastest combination: expected.
- Note for the MBU: spends many iterations in the dual ratio loops.
- Hybrid MOSV is the most efficient among the theoretically finite index selection rules.
- The MBU takes less iteration in average but the more time consuming iterations often pay off.

## 4.2 Iteration and multiplicity of choice

The question arises, what was the level of freedom in choosing the incoming variable in the case of the flexible index selection rules and if it exhibits the expected correlation with efficiency. The sum of this freedom will be called multiplicity in the next tableaus.

In table 10 and 4.2, 'I' stands for Iteration, 'MP' the multiplicity, 'S' and 'M' the simplex and MBU-simplex algorithms respectively. The multiplicity is the sum of the possible index selection choices added together for all iterations and for all problems.

| I | Most negative | Minimal index | LIFO | MOSV | Hybrid LIFO | Hybrid MOSV | SUM |
|---|---|---|---|---|---|---|---|
| **S** | 265k | 989k | 757k | 1 016k | 626k | **341k** | 2 981k |
| **M** | 588k | 442k | 504k | 581k | 438k | **429k** | 3 994k |

Table 10: Iteration counts in thousand iterations.

| MP | Most negative | Minimal index | LIFO | MOSV | Hybrid LIFO | Hybrid MOSV | SUM |
|---|---|---|---|---|---|---|---|
| **S** | 369M | | 239M | 240M | 223M | **251M** | 1 322M |
| **M** | 98M | | 98M | 98M | 89M | 92M | 475M |

The expected correlation between speed and level of multiplicity is apparent, supporting the benefits of flexible rules. The MBU appears to reduce flexibility faster, due to the larger number of **s** updates through the dual ratio test, which makes the flexible index selection rules more rigid much quicker than for the simplex; especially in the case of the LIFO rule.

Figure 3 plots the connection between multiplicity and total solution time (as presented in Table 11 ). The horizontal axis shows the time, while the vertical one the multiplicity.

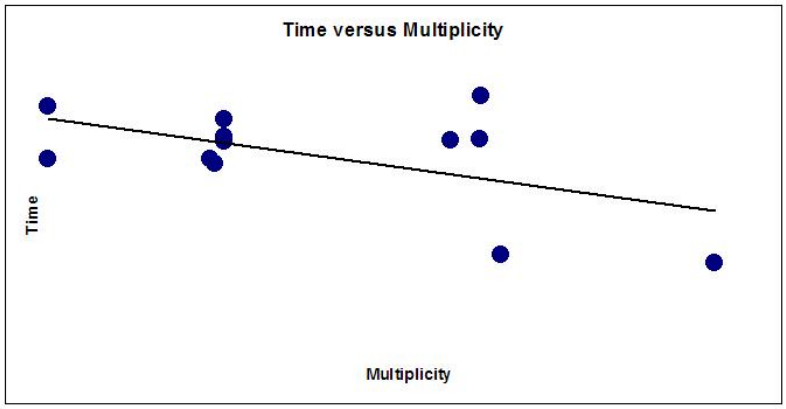


Figure 3: Connection between multiplicity and total solution time.

Turning the previous observation around, we conclude that the longer it took to solve the problems, the more rigid index selection rule was applied.

| | | | | | |
|---|---|---|---|---|---|
| Time | 1 017 | 1 100 | 2 036 | 2 083 | ... |
| Multiplicity | 369 | 250 | 92 | 89 | ... |

Table 11: Ordered set of total time and multiplicities.

### 4.3 Numerical results on selection of industrial problems

Detailed example run statistics are provided on a set of assembly line balancing and workforce skill balancing problems, see [4] and [13] respectively. As with the Miplib sets, only the linear part of the models were considered. These problems provide an insight to the solution of simpler, but realistic problems. This set contains 11 problems. All algorithm and index selection rule combination solved all problems, except for SALBP-1-ESC-3LEV problem on which all combination of the MBU method failed due to numerical issues, incorrectly declaring the problem infeasible.

On these problems, the increase in sizes due to the transformation to the standard form were 1015%, 130% and 59% in the number of rows, columns and elements of the matrix, respectively.

Table 12 collects the relevant run statistics. For each model and algorithm combinations, 3 numbers are presented: iteration, multiplicity and run time; with the exception of the for the first 9 problems, all running times are below 1 second and were omitted from the table.

## 5 Concluding remarks

We have demonstrated that the flexible index selection rules could be a viable alternative in practice when cycling occurs possibly from the (removal of) auto-perturbation itself. Another interesting field of application would be to use these flexible rules in exact arithmetic implementations. The greedy approach in column selection is fastest and the flexible index selection rules can successfully exploit such strategies and provide a significant performance improvement over the more rigid rules - while maintaining theoretical finiteness as well.

We have also demonstrated that the MBU algorithm can be a practically competitive alternative to the traditional (primal) simplex method, as it is demonstrated by Table 3 and Table 4; their theoretically finite versions are comparable both in terms of iteration counts and in terms of solution times, as summarized in Table 8 and Table 9.

## 6 Further research

Some algorithmic concepts like the direct handling of both lower and upper bounded variables, the handling of range and equality constraints or free variables could be undoubtedly be implemented in both algorithms in such a way that their presence do not deteriorate the running times in favor of either methods. While we would expect that such extensions would not change the analysis of this paper, it would make a larger set of problems addressable by the implementations.

It would be interesting to test the flexible index selection rules on special problem classes like network problems, that are highly degenerate and numerically stable.

| | MBU Dantzig | MBU Minimal | MBU LIFO | MBU MOSV | MBU H-LIFO | MBU H-MOSV | Simplex Dantzig | Simplex Minimal | Simplex LIFO | Simplex MOSV | Simplex H-LIFO | Simplex H-MOSV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SALBM-1 | 321 | 392 | 332 | 315 | 316 | 380 | 104 | 369 | 389 | 423 | 187 | 127 |
| | 2246 | 0 | 2205 | 2011 | 1747 | 1760 | 5175 | 0 | 5654 | 5637 | 4550 | 4624 |
| SALBM-1 | 319 | 406 | 352 | 353 | 369 | 388 | 175 | 575 | 492 | 479 | 323 | 247 |
| -ESC | 4112 | 0 | 3996 | 3918 | 3938 | 3959 | 11364 | 0 | 8996 | 8920 | 8185 | 8609 |
| SALBM-1 | 465 | 490 | 478 | 463 | 372 | 367 | 160 | 635 | 724 | 674 | 240 | 305 |
| -HSC | 2977 | 0 | 3034 | 3319 | 2645 | 2807 | 9822 | 0 | 8784 | 9079 | 7586 | 8113 |
| SALBM-1 | 536 | 578 | 521 | 541 | 554 | 684 | 187 | 907 | 969 | 1080 | 448 | 519 |
| -LSC | 3566 | 0 | 3718 | 3803 | 3053 | 2861 | 12089 | 0 | 8970 | 9305 | 8830 | 10175 |
| SALBM-2 | 126 | 138 | 143 | 143 | 116 | 116 | 62 | 167 | 166 | 156 | 120 | 110 |
| | 546 | 0 | 868 | 836 | 508 | 549 | 1622 | 0 | 1614 | 1629 | 1336 | 1390 |
| SALBM-2 | 198 | 184 | 173 | 181 | 197 | 195 | 84 | 283 | 226 | 278 | 135 | 149 |
| -ESC | 1134 | 0 | 1208 | 1132 | 978 | 1078 | 3195 | 0 | 2999 | 3015 | 2422 | 2459 |
| SALBM-2 | 154 | 205 | 222 | 219 | 150 | 145 | 96 | 277 | 320 | 322 | 180 | 152 |
| -HSC | 867 | 0 | 1084 | 1055 | 842 | 895 | 3179 | 0 | 2835 | 2863 | 2218 | 2523 |
| SALBM-2 | 250 | 245 | 226 | 214 | 228 | 201 | 74 | 368 | 480 | 425 | 132 | 125 |
| -LSC | 1034 | 0 | 1136 | 1145 | 890 | 959 | 2248 | 0 | 3125 | 3241 | 2285 | 2108 |
| SALBP-1 | 11535 | 22303 | 7515 | 12438 | 15086 | 13630 | 1214 | 11530 | 7750 | 10076 | 7646 | 14577 |
| | 475151 | 0 | 373007 | 365498 | 263814 | 115885 | 940908 | 0 | 1147003 | 1159227 | 976951 | 969663 |
| | 25s | 54s | 17s | 28s | 32s | 29s | 2s | 18s | 13s | 16s | 12s | 25s |
| SALBP-1 | N/A | | | | | | 4374 | 100108 | 35066 | 161847 | 72664 | 68718 |
| -ESC-3LEV | | | | | | | 4375057 | 0 | 1847066 | 1849618 | 1830788 | 2012826 |
| | | | | | | | 10s | 211s | 74s | 351s | 158s | 171s |
| SALBP-2 | 3595 | 4382 | 3876 | 4048 | 4448 | 4245 | 535 | 15405 | 5667 | 14312 | 4924 | 4785 |
| | 52566 | 0 | 51302 | 51592 | 35683 | 36562 | 170252 | 0 | 146070 | 150643 | 132050 | 169991 |
| | 3s | 4s | 3s | 3s | 3s | 3s | 0s | 9s | 3s | 9s | 3s | 3 |

Table 12: Numerical results on a set of industrial problems.

It could also be argued, that as the monotone simplex method applies a dual ratio test, it is not a primal algorithm, and it could be reasonable to include a dual simplex algorithm in the comparisons.

## Acknowledgements

This research has been supported by the TÁMOP-4.2.1./B-09/1/KMR-2010-0002, Hungarian National Office of Research and Technology with the financial support of the European Union from the European Social Fund.

Tibor Illés acknowledges the research support obtained from Strathclyde University, Glasgow under the *John Anderson Research Leadership Program*.

We are grateful to Zsolt Csizmadia from FICO for the technical support related to Xpress.

## References

[1] A. A. Akkeleş, L. Balogh, and T. Illés, *New variants of the criss-cross method for linearly constrained convex quadratic programming*, European Journal of Operational Research, 157/1 (2004), pp 74–86.

[2] K. Anstreicher and T. Terlaky, *A monotonic build-up simplex algorithm for linear programming*, Operation Research, 42 (1994), pp. 556–561.

[3] D. Avis and V. Chvatal, *Notes on bland's rule*, Mathematical Programming Study, 8 (1978), pp. 24–34.

[4] E.H. Bowman, *Assembly Line Balancing by Linear Programming*, Operation Research, 8 (1960), pp. 385–389.

[5] V. Chvátal, *Linear programming*, W. H. Freeman and Company, New York, 1983.

[6] Z. Csizmadia, *New pivot based methods in linear optimization, and an application in petroleum industry*, PhD Thesis, Eötvös Loránd University of Sciences, 2007. Available at www.cs.elte.hu/∼csisza.

[7] Z. Csizmadia, T. Illés, and A. Nagy, *The s-monotone index selection rules for pivot algorithms of linear programming*, European Journal of Operation Research, 221 (2012), pp. 491–500.

[8] G. B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, N.J., 1963.

[9] FICO XPRESS, *Xpress-Optimizer Reference Manual*, Available at http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx. 2009. June 3.

[10] T. Illés and K. Mészáros, *A new and constructive proof of two basic results of linear programming*, Yugoslav Journal of Operations Research 11/1 (2001), pp. 15–30.

[11] T. Illés and A. Nagy, *Lineáris programozási programok tesztelése*, IF2008 Konferencia, Debrecen, Hungary, 2008.

[12] E. Klafszky and T. Terlaky, *The role of pivoting in proving some fundamental theorems of linear algebra*, Linear Algebra and its Applications, 151 (1991), pp. 97–118.

[13] T. Koltai and V. Tatay, *Formulation of simple workforce skill constraints in assembly line balancing models*, Periodica Polytechnica, Social and Management Sciences, 19/1 (2011), pp. 43–50.

[14] I. Maros, *Computational techniques of the simplex method*, International Series in Operations Research & Management Science, Kluwer Academic Publishers, Boston, MA, 2003.

[15] *Miplib database*. Available at http://miplib.zib.de/. 2012. July 07.

[16] K.G. Murty, *Linear and combinatorial programming*, John Wiley & Sons,Inc., New York, 1976.

[17] A. Nagy, *Új típusú pivot módszerek numerikus összehasonlítása a lineáris programozásban*, MSc Thesis, Eötvös Loránd University of Sciences, 2007. Available at http://people.inf.elte.hu/parad/Magamrol/NagyADiplomamunka.pdf.

[18] *NETLIB database*. Available at http://www.netlib.org/lp/data/. 2010. November 3.

[19] F. Ordónez and R. M. Freund, *Computational experience and the explanatory value of condition measures for linear optimization*, SIAM J. Optimization, 14 (2003), pp. 307–333.

[20] *Source code of the implementations used in this paper*. Available at http://bolyai.cs.elte.hu/opres/orr/SourceCodes.htm.

[21] T. Terlaky, *A véges criss-cross módszer lineáris programozási feladatok megoldására*, Alkalmazott Matematikai Lapok, 10/3-4 (1984), pp. 289–296.

[22] T. Terlaky and S. Zhang, *Pivot rules for linear programming: a survey on recent theoretical developments*, Annals of Operations Research, 46/47(1-4) (1993), pp. 203–233.